UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES
PROGRAMA DE INGENIERIA DE SISTEMAS

ASIGNATURA:	**INGENIERIA DEL SOFTWARE II**
CODIGO:	SIS602
MODALIDAD:	PRESENCIAL TEORICO
INTENSIDAD:	**4** HORAS TEORICAS / SEMANA.
PREREQUISITOS:	INGENIERIA DEL SOFTWARE I Y LAB DE INGENIERIA DEL SOFTWARE I,
CO-REQUISITOS:	LABORATORIO DE INGENIERIA DEL SOFTWARE II
AREA:	INGENIERIA APLICADA
CREDITOS:	3

OBJETIVO GENERAL

Proporcionar al estudiante el espacio de conocimiento, conceptualización e integración de conocimientos para que el aprendiz desarrolle habilidades ingenieriles efectivas relacionadas con la construcción de software de calidad y orientado a objetos; actuando dentro del marco de trabajo de un proceso de desarrollo (guiado por casos de uso, iterativo e incremental y centrado en la arquitectura), al mismo tiempo que promueve: las buenas prácticas, el diseño con heurísticas y patrones y finalmente la valoración de la calidad del producto software mediante métricas orientadas a objetos. En este orden de ideas, esta asignatura está concebida como un espacio reflexivo e integrador donde el aprendiz, no solo adquiere los elementos necesarios para abordar y dar solución a una amplia diversidad de situaciones problémicas en el desarrollo realista de software actual sino que también analiza las implicaciones derivadas de sus decisiones cotidianas en la ejecución de cualquier proyecto de desarrollo de software.

OBJETIVOS ESPECIFICOS

Al finalizar el curso, el estudiante estará en capacidad de:

1. Construir software orientado a objetos basado en la arquitectura documentada (modelos de análisis y diseño realizados en notación UML) en el marco de un proceso de desarrollo ágil.

2. Aplicar heurísticas y patrones para el diseño de software orientado a Objetos con el propósito de enriquecer la expansibilidad, robustez y mantenibilidad del producto software.

3. Aplicar patrones de diseño orientados hacia la construcción de un Esquema (Framework) de Persistencia usando el paradigma orientado a objetos.

4. Fomentar constantemente la reflexión sobre el "Porque hacemos Software como lo hacemos?" en el contexto del Rol del Ingeniero de Sistemas enmarcado en el aporte que este último hace a la sociedad.

METODOLOGIA

1. El alumno adquirirá los conocimientos básicos a través de clases magistrales demostrativas que integran diversos recursos audio-visuales.

2. El alumno desarrollará talleres dirigidos en clase que le ayudarán a llevar a la práctica los conocimientos teóricos adquiridos.

3. El alumno profundizará sus conocimientos con temas complementarios y trabajos de investigación.

CONTENIDO

1. INTRODUCCIÓN Y MOTIVACIÓN A LA INGENIERÍA DEL SOFTWARE II: [4 Horas]

1.1. Perfil y Propósito del Ingeniero de Sistemas.
1.2. Conceptos Generales de los Sistemas.
1.3. El Ingeniero de Sistemas y su Aporte a la Sociedad desde la perspectiva de la ISW.
1.4. Preocupaciones fundamentales de la Industria del Software.
1.5. Buenas y Malas prácticas que contribuyen a mejorar o empeorar la calidad del Producto.
1.6. Principios del Desarrollo Incremental.

2. DESARROLLO DEL CASO DE ESTUDIO-Iteración 1: [16 Horas]

2.1. Ingeniería de Requisitos

2.1.1. Presentación del Caso de Estudio.
2.1.2. Elaboración del Diagrama de Casos de Uso: Propósito y Elementos de Notación.
2.1.3. Elaboración del Diagrama de Casos de Uso: Heurísticas para su Construcción.
2.1.4. El Caso de Uso Especial: Inicio
2.1.5. El Caso de Uso CRUD-Q: Registro, Actualización, Eliminación y Consulta.
2.1.6. Heurísticas para la redacción de casos de uso: El Caso de Uso en Alto Nivel.

2.2. Análisis Orientado a Objetos

2.2.1. Elaboración del Diagrama de Clases: Propósito y Elementos de Notación.
2.2.2. Elaboración del Diagrama de Clases: Heurísticas para su Construcción.
2.2.3. Uso del Patrón OID: Identificador de Objetos Persistentes.
2.2.4. Uso del Patrón GRASP Alta Cohesión: Ejemplos y Aplicación.
2.2.5. Uso del Patrón GRASP Bajo Acoplamiento: Ejemplos y Aplicación.
2.2.6. Cálculo de la Métrica Coupling Between Objects (CBO): Acoplamiento entre Objetos.
2.2.7. Cálculo de la Métrica Coupling Factor (CF): Factor de Acoplamiento.
2.2.8. Cálculo de la Métrica Number of Classes (NUMCLASS): Número de Clases
2.2.9. Cálculo de la Métrica Number of Operations (NUMOPS): Número de Operaciones

2.3. Diseño Orientado a Objetos:

2.3.1. Elaboración del Diagrama de Colaboración: Propósito y Elementos de Notación.
2.3.2. Elaboración del Diagrama de Colaboración: Heurísticas para su Construcción.
2.3.3. Diagramando las Operaciones del Cuso CRUD-Q: 4 Soluciones y Reflexión.
2.3.4. Reflexiones sobre cada solución: Violaciones al Paradigma OO, Optimización.
2.3.5. Uso del Patrón GRASP: Creador.
2.3.6. Cálculo de la Métrica Number of Objects (NUMOBJECTS): Número de Objetos.
2.3.7. Cálculo de la Métrica Number of Messages (NUMMSGS): Número de Mensajes.
2.3.8. Elaboración del Diagrama de Paquetes: Propósito y Elementos de Notación.
2.3.9. Patrón GRASP: Capas de la Arquitectura.
2.3.10. Patrón GRASP: Controlador de Fachada.

2.4. Implementación Orientada a Objetos: Mapeo a Código.

2.4.1. Heurísticas para la construcción de Código orientado a Objetos: Nombrado y Estilo
2.4.2. Regla 1**:** Mapeo de Mensajes Entrantes.
2.4.3. Regla 2: Mapeo de Mensajes Salientes.
2.4.4. Cálculo de la Métrica Number of Lines of Code (NUMLOC) : Líneas de Código

3. CASO ESTUDIO-Iteración 2: [20 Horas]

3.1. Ingeniería de Requisitos

3.1.1. Elaboración del Diagrama de Casos de Uso: Agregando más Casos de Uso.
3.1.2. Elaboración del Diagrama de Casos de Uso: Relacionando Casos de Uso.
3.1.3. Heurísticas para la redacción de casos de uso: El Caso de Uso Expandido.

3.2. Análisis Orientado a Objetos

3.2.1. Uso del Patrón GRASP Separación Modelo-Vista: Ejemplos y Aplicación.
3.2.2. Uso del Patrón GRASP No Hables con Extraños: Ejemplos y Aplicación.
3.2.3. Elaboración del Diagrama de Clases: Agregando más Clases.
3.2.4. Determinación de la Visibilidad: 4 tipos de Visibilidad.
3.2.5. Heurística 1: Agregando Atributos Propios.
3.2.6. Heurística 2**:** Agregando Atributos Asociativos Únicos.
3.2.7. Heurística 3: Agregando Atributos Asociativos En Colección.
3.2.8. Heurística 4**:** Agregando Atributos Derivables.
3.2.9. Heurística 5**:** Agregando Métodos Constructores.
3.2.10. Heurística 6**:** Agregando Método Destructor.
3.2.11. Heurística 7**:** Agregando Métodos Accesores de Atributo Propio.
3.2.12. Heurística 8**:** Agregando Métodos Accesores Asociativos.
3.2.13. Heurística 9**:** Agregando Métodos Accesores Asociativos Utilitarios.
3.2.14. Heurística 10**:** Agregando Métodos Mutadores de Atributo Propio.
3.2.15. Heurística 11**:** Agregando Métodos Mutadores Asociativos.
3.2.16. Heurística 12**:** Agregando Métodos Mutadores Disociativos.
3.2.17. Heurística 13**:** Agregando Métodos Registradores.
3.2.18. Heurística 14**:** Agregando Métodos Actualizadores.
3.2.19. Heurística 15**:** Agregando Métodos Eliminadores.
3.2.20. Heurística 16**:** Agregando Métodos Consultores.
3.2.21. Heurística 18**:** Agregando Métodos Derivables.
3.2.22. Re-Cálculo de las Métricas: CBO, COF, NUMCLASS y NUMOPS.

3.3. Diseño Orientado a Objetos:

3.3.1. Diagramando las Operaciones del Cuso CRUD-Q: Usando el Patrón GRASP Experto.
3.3.2. Heurísticas 5 a 18: Diagramas de Colaboración.
3.3.3. Patrón: No hables con Extraños.
3.3.4. Re-Cálculo de las Métricas NUMOBJECTS, NUMMSGS,

3.4. Implementación Orientada a Objetos: Mapeo a Código.

3.4.1. Refactorización del Código: Usando Heurísticas de la 1 a la 18.
3.4.2. Re-Cálculo de la Métrica Number of Lines of Code (LOC) : Número de Líneas de Código

4. CASO ESTUDIO-Iteración 3: [24 Horas]

4.1. Agregando la Capa de Persistencia

4.1.1. Fundamentos del Diseño de FrameWorks.
4.1.2. Requerimientos de un Framework e Ideas Básicas.
4.1.3. Aplicando el Patrón: Representación de Objetos como Tablas.
4.1.4. Aplicando el Patrón: Intermediario de Bases de Datos.
4.1.5. Aplicando el Patrón: Método Plantilla.
4.1.6. Aplicando el Patrón: Administración de Caché.
4.1.7. Aplicando el Patrón: Agente Virtual y Puente.
4.1.8. Aplicando el Patrón: Instanciación de Objetos Complejos.
4.1.9. Aplicando el Patrón: Singleton.
4.1.10. Aplicando el Patrón: Polimorfismo.
4.1.11. Heurísticas de Representación: Diagrama de Clases como Diagrama E-R.
4.1.12. Manejo de Transacciones.
4.1.13. Búsqueda de Objetos desde el Almacenamiento Persistente.

4.2. Determinando Costos y Esfuerzo del Desarrollo

4.2.1. Coste y Esfuerzo usando métricas de Tamaño.
4.2.2. Coste y Esfuerzo usando métricas de Complejidad.
4.2.3. Estimación de la Facilidad de Mantenimiento usando métricas de Acoplamiento.
4.2.4. Estimación de la Facilidad de Comprensión usando métricas de Cohesión.
4.2.5. Estimación de la Facilidad de Re-uso usando métricas de Herencia.
4.2.6. Reflexiones Finales: Desarrollo Caótico Vs. Desarrollo Disciplinado.

PLANEACIÓN CON RESPECTO AL LABORATORIO

Semana	Tema Asignatura Teórica	Práctica Laboratorio
1	Capitulo 1	Sin Práctica
2	Capítulo 2	Práctica 1
3	Capítulo 2	Práctica 1
4	Capítulo 2	Práctica 1
5	Capítulo 2	Práctica 2
6	Capítulo 3	Práctica 3
7	Capítulo 3	Práctica 3
8	Capítulo 3	Práctica 3
9	Capítulo 3	Práctica 3
10	Capítulo 3	Práctica 4
11	Capítulo 4	Práctica 5
12	Capítulo 4	Práctica 5
13	Capítulo 4	Práctica 5
14	Capítulo 4	Práctica 5
15	Capítulo 4	Práctica 5

EVALUACIONES

Se realizarán tres (3) evaluaciones de la siguiente forma:

NUMERO	%	COMPONENTES
Primer Parcial	35%	Parcial Escrito 70% Quices, Talleres 30%
Segundo Parcial	35%	Parcial Escrito 70% Quices, Talleres 30%
Tercer Parcial	30%	Parcial Escrito 70% Quices, Talleres 30%

BIBLIOGRAFÍA

- Pressman, Roger. Ingeniería del Software, Un Enfoque Práctico. McGraw-Hill. 1998.
- Craig, Larman. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall. 1999.
- UML Ed. Addison Wesley
- http://www.devx.com/
- http://www.sei.cmu.edu/
- Ivar Jacobson, Grady Booch y James, Rumbaugh. The Unified Software Development Process. Rational Software Corporation. Addison Wesley, 1999. ISBN:0-201-57169-2. 463 Págs.
- Arthur Riel. Object Oriented Design Heuristics. Addison Wesley.

LECTURAS COMPLEMENTARIAS

- M. Fowler, Diagramas de clase: conceptos avanzados, Adisson Wesley Longman, *UML gota a gota,* (México: Estado de México, 1999).
- M. Fowler, Los casos de uso, Adisson Wesley Longman, *UML gota a gota,* (México: Estado de México,1999).
- Watts S. Managing the Software Process. Humpherey. Addison-Wesley. 1989.
- Jones, Capers. Applied Software Measurement: Assuring Productivity and Quality. McGraw-Hill. 1991.
- Watts S. Humphrey.Introduction to the Team Software Process.
- Moreno Jorge. Proceso Agil de Desarrollo en Comunidad. Agile-DISOP.
- Riel Arthur. Heurísticas de Diseño

SitioWeb de la Asignatura

- **http://pis.unicauca.edu.co/moodle/course/view.php?id=352**